

Building Deep Networks on Grassmann Manifolds

Zhiwu Huang, Jiqing Wu, Luc Van Gool
Computer Vision Laboratory, D-ITET, ETH Zurich
{zhiwu.huang, jwu, vangool}@vision.ee.ethz.ch

Abstract

Representing the data on Grassmann manifolds is popular in quite a few image and video recognition tasks. In order to enable deep learning on Grassmann manifolds, this paper proposes a deep network architecture which generalizes the Euclidean network paradigm to Grassmann manifolds. In particular, we design full rank mapping layers to transform input Grassmannian data into more desirable ones, exploit orthogonal re-normalization layers to normalize the resulting matrices, study projection pooling layers to reduce the model complexity in the Grassmannian context, and devise projection mapping layers to turn the resulting Grassmannian data into Euclidean forms for regular output layers. To train the deep network, we exploit a stochastic gradient descent setting on manifolds where the connection weights reside on, and study a matrix generalization of backpropagation to update the structured data. We experimentally evaluate the proposed network for three computer vision tasks, and show that it has clear advantages over existing Grassmann learning methods, and achieves results comparable with state-of-the-art approaches.

1. Introduction

In a variety of computer vision and machine learning domains, deep neural networks have impressively surpassed conventional shallow learning models. The great success of deep neural networks is mainly attributed to the power of performing non-linear computations on the visual data, and the effectiveness of the gradient-descent training procedure based on backpropagation.

By leveraging the successful deep learning paradigm of traditional neural networks, increasing new neural networks [10, 27, 8, 45, 3, 35, 29] have been built over general non-Euclidean domains in recent years. For instance, [10] proposed a spectral variant of convolution networks to graphs through Graph Fourier Transform, which is in turn defined via a generalization of the Laplacian operator on the grid to graph Laplacian. More recently, [35] presented a deep learning approach on spatio-temporal graphs by treating an

arbitrary spatio-temporal graph as a recurrent neural network mixture that is feedforward, differential and jointly trainable. In [29], to deeply learn appropriate features on the manifolds of symmetric positive definite matrices, a deep network was built with some spectral layers, which can be trained by a variant of backpropagation.

In this paper, we focus on studying how to build a deep neural network architecture on Grassmann manifolds, where the data have become a core representation of many computer vision techniques. For example, as a basic form of Grassmannian data, linear subspace has proven beneficial for matching image sets due to its ability to encode the useful second-order information of one set of images from a certain object class [20, 24, 23, 31]. For video visual recognition, linear dynamic system (i.e., linear subspace) of autoregressive and moving average model has shown its great power to model dynamics in spatio-temporal processing [12, 52, 54]. In addition, Grassmannian data also appear in image restoration [56], chromatic noise filtering [51] and domain adaptation [18].

The popular applications of Grassmannian data motivate us to build a deep neural network architecture for Grassmannian representation learning. For this purpose, the new network architecture is designed to take Grassmannian data directly as input, and learns new favorable Grassmannian data that are able to improve the final visual tasks. In other words, the new network aims to deeply learn Grassmannian data on their underlying Riemannian manifolds in an end-to-end learning architecture. In summary, the main contributions of this paper are two-fold:

- We explore a new network architecture in the context of Grassmann manifolds, where it has not been possible to apply deep neural networks. Thanks to this network, a new direction of deeply learning Grassmannian data has been opened up.
- We study a procedure to train the new network. In the procedure, we derive an update rule for the connection weights on a specific type of Riemannian manifolds, and exploit matrix backpropagation to update the structured data within typical matrix factorizations.

2. Background

2.1. Grassmannian Geometry

A Grassmann manifold $\mathcal{G}(q, D)$ is a $q(D - q)$ dimensional compact Riemannian manifold, which is the set of q -dimensional linear subspaces of the \mathbb{R}^D . Thus, each point on $\mathcal{G}(q, D)$ is a linear subspace that is spanned by its orthonormal basis matrix \mathbf{X} of size $D \times q$ such that $\mathbf{X}^T \mathbf{X} = \mathbf{I}_q$, where \mathbf{I}_q is the identity matrix of size $q \times q$.

On a Grassmannian $\mathcal{G}(q, D)$, points are connected via smooth curves. The geodesic distance between two points is defined as the length of the shortest curve connecting them on the Riemannian manifold. The shortest curve and its length are named geodesic and geodesic distance, respectively. On the Grassmannian, the geodesic distance between two points \mathbf{X}_1 and \mathbf{X}_2 is computed by

$$d_g(\mathbf{X}_1, \mathbf{X}_2) = \|\Theta\|_2, \quad (1)$$

where Θ is the vector of principal angles between \mathbf{X}_1 and \mathbf{X}_2 . For its computation, readers are referred to [16].

One of the most popular approaches to approximate the true Grassmannian geodesic distance is the projection mapping framework $\Phi(\mathbf{X}) = \mathbf{X}\mathbf{X}^T$ proposed by [16]. As the projection $\Phi(\mathbf{X})$ is a $D \times D$ symmetric matrix, a natural choice of inner product is $\langle \mathbf{X}_1, \mathbf{X}_2 \rangle_\Phi = \text{tr}(\Phi(\mathbf{X}_1)^T \Phi(\mathbf{X}_2))$. The inner product induces a distance named projection metric:

$$d_p(\mathbf{X}_1, \mathbf{X}_2) = 2^{-1/2} \|\mathbf{X}_1 \mathbf{X}_1^T - \mathbf{X}_2 \mathbf{X}_2^T\|_F, \quad (2)$$

where $\|\cdot\|_F$ indicates the matrix Frobenius norm. As proved in [23], the projection metric can approximate the true geodesic distance Eqn.1 up to a scale of $\sqrt{2}$.

2.2. Grassmann Learning

To perform discriminant learning on Grassmann manifolds, many works [20, 19, 11, 24, 23, 22, 37] embed the Grassmannian into a Euclidean space. This can be achieved either by tangent space approximation of the underlying manifold, or by exploiting a positive definite kernel function to embed the manifold into a reproducing kernel Hilbert space. In both of such two cases, any existing Euclidean technique can then be applied to the embedded data, since Hilbert spaces respect Euclidean geometry. For example, [20] first embeds the Grassmannian into a high dimensional Hilbert space, and then applies traditional Fisher analysis method. Obviously, most of these methods are limited to the Mercer kernels and hence restricted to use only kernel-based classifiers. Moreover, their computational complexity increases steeply with the number of training samples.

More recently, a new learning scheme was proposed in [31] to perform a geometry-aware dimensionality reduction from the original Grassmann manifold to a lower-dimensional, more discriminative Grassmann manifold.

This could better preserve the original Riemannian data structure, which commonly leads to more favorable classification performances as studied in classical manifold learning. While [31] has reached some success, it merely adopts a shallow learning scheme on Grassmann manifolds, which is still far away from the best solution for the problem of representation learning on non-linear manifolds. Accordingly, this paper attempts to open up a possibility of deep learning on Grassmannians.

3. Grassmann Network Architecture

The proposed Grassmann network (GrNet) performs deep learning on Grassmannians for general visual recognition problems. As classical convolutional networks (ConvNets), the proposed network also designs fully connected convolution-like layers and normalization layers, named full rank mapping (FRMap) layers and orthogonal re-normalization (ReOrth) layers respectively. Inspired by the geometry-aware manifold learning idea [31], the FRMap layers are proposed to firstly perform transformations on input orthonormal matrices to generate new matrices by adopting a full rank mapping scheme. Then, the ReOrth layers are developed to normalize the output matrices of the FRMap layers so that they can become valid orthonormal matrices to form a Grassmann manifold. With a specific non-linear matrix decomposition, to some extent, the ReOrth layers serve as a role of introducing the non-linearity to the network as well. As traditional pooling layers, we also study special pooling layers for the Grassmannian data to reduce the network complexity. As the pooling is performed on the projection matrix form of the orthonormal matrices, we call this kind of layer as ProjPooling layers. To enable the regular Euclidean layers such as softmax layers to work for the GrNet, we also exploit projection mapping (ProjMap) layers to map the resulting orthonormal matrices into projection matrices that lie in Euclidean space. The proposed Grassmann network is illustrated in Fig.1.

3.1. FRMap Layer

The GrNet is basically to learn more compact and discriminative Grassmannian data for better classification in an end-to-end learning manner. For this purpose, we design the FRMap layers to first transform the input orthonormal matrices to new matrices by a linear mapping f_{fr} as

$$\mathbf{X}_k = f_{fr}^{(k)}(\mathbf{X}_{k-1}; \mathbf{W}_k) = \mathbf{W}_k \mathbf{X}_{k-1}, \quad (3)$$

where $\mathbf{X}_{k-1} \in Gr(q, d_{k-1})$ is the input orthonormal basis matrix of the k -th layer, $\mathbf{W}_k \in \mathbb{R}_*^{d_k \times d_{k-1}}$, ($d_k < d_{k-1}$) is the transformation matrix (connection weights) that is basically required to be a row full-rank matrix, $\mathbf{X}_k \in \mathbb{R}^{d_k \times q}$ is the resulting matrix. Unfortunately, except the case \mathbf{W} is an

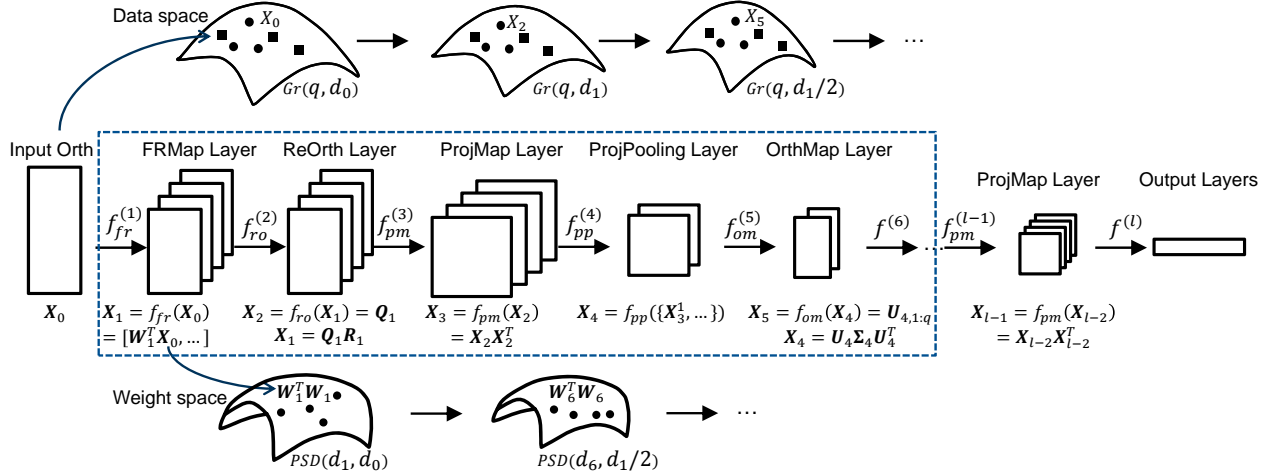


Figure 1. Conceptual illustration of the proposed Grassmann Network (GrNet) architecture. The space of output data from each ReOrth/OrthMap layer is one Grassmann manifold, while the weight conjugate spaces of the FRMap layers correspond to PSD manifolds.

orthogonal matrix, $\mathbf{W}_k \mathbf{X}_{k-1}$ is not generally an orthonormal basis matrix. To tackle this problem, we exploit a normalization strategy of QR decomposition in the following ReOrth layer. In addition, as classical deep networks, multiple projections $\{\mathbf{W}_k^1, \dots, \mathbf{W}_k^m\}$ per FRMap layer can be applied on each input orthonormal matrix as well, where m is the number of transformation matrices.

As the weight space $\mathbb{R}_*^{d_k \times d_{k-1}}$ of full-rank matrices on each FRMap layer is a non-compact Stiefel manifold where the distance function has no upper bound, optimizing on the manifold directly is infeasible. To handle this problem, one possible solution is imposing orthogonality constraint on the transformation matrix \mathbf{W}_k so that it resides on a compact Stiefel manifold $St(d_k, d_{k-1})$. Obviously, such orthogonal solution space is smaller than the original solution space $\mathbb{R}_*^{n \times m}$, making the optimization theoretically yield suboptimal solution of \mathbf{W}_k . To achieve more faithful solution of the full-rank transformation matrix \mathbf{W}_k , following [31, 30], we alternatively choose to do the optimization over the manifolds $PSD(d_k, d_{k-1})^1$ of the conjugates $\mathbf{P}_k = \mathbf{W}_k^T \mathbf{W}_k$, which are actually positive semidefinite (PSD) matrices. As studied in [30], since \mathbf{P}_k can be easily parametrized by \mathbf{W}_k , optimizing on the PSD manifold actually pursues optimal \mathbf{W}_k directly.

3.2. ReOrth Layer

To make the resulting matrices reside on a valid Grassmann manifold, the ReOrth layer normalizes the matrix \mathbf{X}_{k-1} to \mathbf{X}_k so that the columns of \mathbf{X}_k are orthonormal. Specifically, we perform QR decomposition on \mathbf{X}_{k-1} :

$$\mathbf{X}_{k-1} = \mathbf{Q}_{k-1} \mathbf{R}_{k-1}, \quad (4)$$

¹A PSD manifold $PSD(d_k, d_{k-1})$ is the set of d_k -rank positive semidefinite matrices of size d_{k-1} [7, 38, 46, 31, 30].

where $\mathbf{Q}_{k-1} \in \mathbb{R}^{d_{k-1} \times q}$ is the orthonormal matrix composed by the first q columns, and $\mathbf{R}_{k-1} \in \mathbb{R}^{q \times q}$ is the invertible upper-triangular matrix. Since \mathbf{R} is invertible and \mathbf{Q} is orthonormal, we can make \mathbf{X}_k become an orthonormal basis matrix by normalizing \mathbf{X}_{k-1} in the k -th layer as:

$$\mathbf{X}_k = f_{ro}^{(k)}(\mathbf{X}_{k-1}) = \mathbf{X}_{k-1} \mathbf{R}_{k-1}^{-1} = \mathbf{Q}_{k-1}. \quad (5)$$

In the context of ConvNets, [13, 36, 48, 17, 25] have presented various nonlinear activation functions, e.g., rectified linear units (ReLU), to improve discriminative performance. Accordingly, exploiting this kind of layers to introduce the non-linearity to the domain of the proposed GrNet is also necessary. In the light of this, to some extent, the function Eqn.5 also takes a role of performing a non-linear operation based on the QR factorization.

3.3. ProjMap Layer

The ProjMap layer is designed to perform Riemannian computing on the resulting orthonormal matrices on one Grassmannian. As well-studied in [16, 20, 19, 24, 23, 31], the projection metric is one of the most popular Grassmannian metrics, and is able to endow the specific Riemannian manifold with an inner product structure so that the manifold is reduced to a flat space. In the flat space, classical Euclidean computations can be applied to the projection domain of orthonormal matrices. Formally, we apply the projection mapping [16] on a orthonormal matrix in the k -th layer with the function f_{pm} being defined as

$$\mathbf{X}_k = f_{pm}^{(k)}(\mathbf{X}_{k-1}) = \mathbf{X}_{k-1} \mathbf{X}_{k-1}^T. \quad (6)$$

As for other Riemannian computations on Grassmann manifolds, please refer to [55, 40, 16, 50, 2, 26, 23] for more detailed discussions about their properties.

3.4. ProjPooling Layer

It is known that classical pooling layers with max, min and mean pooling functions reduce the sizes of the representations to lower the model complexity and improve the regular ConvNets. Motivated by this, we also design pooling layers tailored for the feature maps of Grassmannian data. Without loss of generality, we here study the mean pooling functions for the Grassmannian points. As far as we investigate, there exist some approaches [1, 15, 49, 44] to compute mean points on Grassmannians. One of the most popular methods is to compute the Karcher mean [1], which is the intrinsic mean on Grassmann manifolds by minimizing the geodesic distances of the difference defined in Eqn.1. Unfortunately, the Karcher mean algorithms require very expensive iterative exponential and logarithm maps to move the data to and from the tangent space. To speed up the computation, [49] proposed an alternative to the Karcher mean by minimizing the Frobenius norm squared of the difference of the projection maps of Grassmannian data.

Inspired by the idea [49], we propose three layers to implement the average pooling for Grassmannian data. In particular, the Grassmannian data are first mapped to the space of projection matrices by the ProjMap layer presented before. As the resulting projection matrices \mathbf{X}_{k-1}^i are Euclidean, we then design a regular mean pooling layer for them. Lastly, we devise an orthogonal map (OrthMap) layer to map the mean projection matrix back to the underlying Grassmann manifold. Formally, the functions for the last two layers (i.e., ProjPooling and OrthMap) are defined as

$$\mathbf{X}_k = f_{pp}^{(k)}(\{\mathbf{X}_{k-1}^1, \dots, \mathbf{X}_{k-1}^n\}) = \frac{1}{n} \sum_i^n \mathbf{X}_{k-1}^i, \quad (7)$$

$$\mathbf{X}_{k+1} = f_{om}^{(k)}(\mathbf{X}_k) = \mathbf{U}_{k-1,1:q}, \quad (8)$$

where $\mathbf{U}_{k-1,1:q}$ is the first q leadest eigenvectors achieved by symmetric eigenvalue (EIG) computation on the input projection matrices $\mathbf{X}_k = \mathbf{U}_{k-1} \mathbf{\Sigma}_{k-1} \mathbf{U}_{k-1}^T$, and n is the number of instances for the pooling. Note that the instances for pooling could be multiple projection matrices or multiple elements within an $n \times n$ patch located in one projection matrix. In the experiments, we will further study these two different pooling schemes.

3.5. Output Layers

After applying the ProjMap layer, the outputs (i.e., the projection matrices) lie in Euclidean space and thus can be transformed into vector forms. Hence, on the top of the ProjMap layer, classical Euclidean network layers can be employed. For instance, the regular fully connected (FC) layer could be utilized after the ProjMap layer. The dimensionality of the filters in the FC layer is typically set to $d_k \times d_{k-1}$, where d_k and d_{k-1} are the class number and the dimensionality of the input vector forms respectively. In the end,

the common softmax layer or softmax log-loss layer can be equipped for the visual classification tasks.

4. Training Grassmann network

As most layers in the GrNet model are expressed with the complex matrix factorization functions, they cannot be simply reduced to a constructed bottom-up from element-wise calculations. In other words, the matrix backpropagation cannot be derived by using traditional matrix that deals element-wise operations in matrix form. As a result, simply using the traditional backpropagation training procedure will break down in the setting. To solve the problem, [29, 33] introduced manifold-valued connection weight update rule and matrix backpropagation (backprop) respectively. Furthermore, the convergence of the stochastic gradient descent (SGD) algorithm on Riemannian manifolds has also been studied well in [9, 6]. Accordingly, we exploit the training procedure for the proposed GrNet upon these two works.

To begin with, we represent the proposed GrNet model with a sequence of successive function compositions $f = f^{(l)} \circ f^{(l-1)} \circ f^{(l-2)} \dots \circ f^{(2)} \circ f^{(1)}$ with a parameter tuple $\mathbf{W} = (\mathbf{W}_l, \mathbf{W}_{l-1}, \dots, \mathbf{W}_1)$, where $f^{(k)}$ and \mathbf{W}_k are the function and the weight matrix respectively for the k -th layer, and l is the number of layers. The loss of the k -th layer can be denoted by a function as $L^{(k)} = \ell \circ f^{(l)} \circ \dots \circ f^{(k)}$, where ℓ is the loss function for the last output layer.

Then, we recall the definition of the matrix backprop and its properties studied in [33]. In particular, [33] exploits a function \mathcal{F} to describe the variations of the upper layer variables with respect to the lower layer variables, i.e., $d\mathbf{X}_k = \mathcal{F}(d\mathbf{X}_{k-1})$. Consequently, a new version of the chain rule for the matrix backprop is defined as

$$\frac{\partial L^{(k)}(\mathbf{X}_{k-1}, y)}{\partial \mathbf{X}_{k-1}} = \mathcal{F}^* \left(\frac{\partial L^{(k+1)}(\mathbf{X}_k, y)}{\partial \mathbf{X}_k} \right), \quad (9)$$

where y is the desired output, $\mathbf{X}_k = f^{(k)}(\mathbf{X}_{k-1})$, \mathcal{F}^* is a non-linear adjoint operator of \mathcal{F} , i.e., $a : \mathcal{F}(b) = \mathcal{F}^*(a) : b$, the matrix inner product $\mathbf{A} : \mathbf{B} = \text{Tr}(\mathbf{A}^T \mathbf{B})$.

In the sequel, we will detail the connection weight update on manifolds and the matrix backprop process through some key layers in the context of the proposed GrNet. For simplicity, we uniformly let $\partial L^{(k)}(\mathbf{X}_{k-1}, y)$ be $\partial L^{(k)}$, \mathbf{Q}_{k-1} be \mathbf{Q} , and \mathbf{R}_{k-1} be \mathbf{R} respectively in the following.

4.1. FRMap Layer

For the weight update in the FRMap layers, we propose a new way of updating the weights appeared in Eqn.3 by exploiting an SGD setting on PSD manifolds. As studied in [2, 29], the steepest descent direction for the corresponding loss function $L^{(k)}(\mathbf{X}_{k-1}, y)$ with respect to \mathbf{W}_k on one Riemannian manifold is the Riemannian gradient $\tilde{\nabla} L_{\mathbf{W}_k}^{(k)}$.

In particular, the parallel transport is first applied to transfer the Euclidean gradient in the tangent space at the anchor point \mathbf{W}_k to the one in the tangent space at the point \mathbf{W}_{k+1} . Then the resulting Euclidean gradient is subtracted to the normal component of the Euclidean gradient $\nabla L_{\mathbf{W}_k}^{(k)}$. After this operation, searching along the tangential direction yields the update in the tangent space of the PSD manifold. Finally, the resulting update is mapped back to the PSD manifold with a retraction operation Γ . For more details about the geometry of PSD manifolds and its retraction operation, readers are referred to [7, 38, 46, 31, 30, 2]. Accordingly, the update of the connection weight \mathbf{W}_k on the PSD manifold respects the following form

$$\tilde{\nabla} L_{\mathbf{W}_k}^{(k)} = \nabla L_{\mathbf{W}_k}^{(k)} - \nabla L_{\mathbf{W}_k}^{(k)} \mathbf{W}_k^T \mathbf{W}_k, \quad (10)$$

$$\mathbf{W}_k^{t+1} = \Gamma(\mathbf{W}_k^t - \lambda \tilde{\nabla} L_{\mathbf{W}_k}^{(k)}), \quad (11)$$

where \mathbf{W}_k^t is the current weight, Γ is the retraction operation, λ is the learning rate, $\nabla L_{\mathbf{W}_k}^{(k)} \mathbf{W}_k^T \mathbf{W}_k$ is the normal component of the Euclidean gradient $\nabla L_{\mathbf{W}_k}^{(k)}$. By employing the conventional backprop, $\nabla L_{\mathbf{W}_k}^{(k)}$ is computed by

$$\nabla L_{\mathbf{W}_k}^{(k)} = \frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k} \frac{\partial f^{(k)}(\mathbf{X}_{k-1})}{\partial \mathbf{W}_k} = \frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k} \mathbf{X}_{k-1}^T. \quad (12)$$

4.2. ReOrth Layer

Actually, the ReOrth layers involve QR decomposition Eqn.4 and the non-linear operation Eqn.5. Firstly, for Eqn.4 we introduce a virtual layer k' , which receives \mathbf{X}_{k-1} as input and produces a tuple (\mathbf{Q}, \mathbf{R}) . Following [34] to deal with the case of outputting a tuple, we apply the new chain rule Eqn.9 with the equations $a : \mathcal{F}(b) = \mathcal{F}^*(a) : b$ and $d\mathbf{X}_k = \mathcal{F}(d\mathbf{X}_{k-1})$ to achieve the update rule for the data:

$$\begin{aligned} & \frac{\partial L^{(k)}}{\partial \mathbf{X}_{k-1}} : d\mathbf{X}_{k-1} \\ &= \mathcal{F}^* \left(\frac{\partial L^{(k')}}{\partial \mathbf{Q}} \right) : d\mathbf{X}_{k-1} + \mathcal{F}^* \left(\frac{\partial L^{(k')}}{\partial \mathbf{R}} \right) : d\mathbf{X}_{k-1} \\ &= \frac{\partial L^{(k')}}{\partial \mathbf{Q}} : \mathcal{F}(d\mathbf{X}_{k-1}) + \frac{\partial L^{(k')}}{\partial \mathbf{R}} : \mathcal{F}(d\mathbf{X}_{k-1}) \\ &= \frac{\partial L^{(k')}}{\partial \mathbf{Q}} : d\mathbf{Q} + \frac{\partial L^{(k')}}{\partial \mathbf{R}} : d\mathbf{R}, \end{aligned} \quad (13)$$

where the two variations $d\mathbf{Q}$ and $d\mathbf{R}$ are derived by the variation of the QR operation $d\mathbf{X}_{k-1} = d\mathbf{Q}\mathbf{R} + \mathbf{Q}d\mathbf{R}$ as:

$$d\mathbf{Q} = \mathbf{S}d\mathbf{X}_{k-1}\mathbf{R}^{-1} + \mathbf{Q}(\mathbf{Q}^T d\mathbf{X}_{k-1}\mathbf{R}^{-1})_{asym}, \quad (14)$$

$$d\mathbf{R} = \mathbf{Q}^T d\mathbf{X}_{k-1} - (\mathbf{Q}^T d\mathbf{X}_{k-1}\mathbf{R}^{-1})_{asym}\mathbf{R}, \quad (15)$$

where $\mathbf{S} = \mathbf{I} - \mathbf{Q}\mathbf{Q}^T$, \mathbf{I} is an identity matrix, $\mathbf{A}_{asym} = \mathbf{A}_{tril} - (\mathbf{A}_{tril})^T$, \mathbf{A}_{tril} extracts the elements below the

main diagonal of \mathbf{A} . For more details to derive Eqn.14 and Eqn.15, readers are referred to the first part of the Appendix.

As derived in the second part of the Appendix, plugging Eqn.14 and Eqn.15 into Eqn.13 gives the partial derivatives of the loss functions for the ReOrth layers:

$$\begin{aligned} \frac{\partial L^{(k)}}{\partial \mathbf{X}_{k-1}} &= \left(\mathbf{S}^T \frac{\partial L^{(k')}}{\partial \mathbf{Q}} + \mathbf{Q} \left(\mathbf{Q}^T \frac{\partial L^{(k')}}{\partial \mathbf{Q}} \right)_{bsym} \right) (\mathbf{R}^{-1})^T \\ &+ \mathbf{Q} \left(\frac{\partial L^{(k')}}{\partial \mathbf{R}} - \left(\frac{\partial L^{(k')}}{\partial \mathbf{R}} \mathbf{R}^T \right)_{bsym} \right) (\mathbf{R}^{-1})^T, \end{aligned} \quad (16)$$

where $\mathbf{A}_{bsym} = \mathbf{A}_{tril} - (\mathbf{A}^T)_{tril}$, \mathbf{A}_{tril} extracts the elements below the main diagonal of \mathbf{A} . $\frac{\partial L^{(k')}}{\partial \mathbf{Q}}$ and $\frac{\partial L^{(k')}}{\partial \mathbf{R}}$ can then be obtained on the function Eqn.5 employed in the ReOrth layers. Specially, its variation becomes $d\mathbf{X}_k = d\mathbf{Q}$. Therefore, the involved partial derivatives with respect to \mathbf{Q} and \mathbf{R} are computed by

$$\frac{\partial L^{(k')}}{\partial \mathbf{Q}} = \frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k}, \quad (17)$$

$$\frac{\partial L^{(k')}}{\partial \mathbf{R}} = 0. \quad (18)$$

4.3. OrthMap Layer

As presented before, the OrthMap layers involve symmetric eigenvalue (EIG) computation. Thus, we adopt the proposition in [33] to calculate the partial derivatives for the eigenvalue computation in the matrix backprop setting.

Proposition 1 Let $\mathbf{X}_{k-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ with $\mathbf{X}_{k-1} \in \mathbb{R}^{D \times D}$, such that $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and $\mathbf{\Sigma}$ owns a diagonal structure. The resulting partial derivative for the EIG layer k' is given by

$$\begin{aligned} \frac{\partial L^{(k)}}{\partial \mathbf{X}_{k-1}} &= \mathbf{U} \left(2 \left(\hat{\mathbf{K}}^T \circ \left(\mathbf{U}^T \frac{\partial L^{(k')}}{\partial \mathbf{U}} \right)_{sym} \right) \right) \mathbf{U}^T \\ &+ \mathbf{U} \left(\frac{\partial L^{(k')}}{\partial \mathbf{\Sigma}} \right)_{diag} \mathbf{U}^T, \end{aligned} \quad (19)$$

where $\hat{\mathbf{K}} = 1/(\sigma_i - \sigma_j), i \neq j; 0, i = j$ (here, σ_i is the diagonal element of $\mathbf{\Sigma}$), and the partial derivatives with respect to $\mathbf{\Sigma}$ and \mathbf{U} in Eqn.8 for the OrthMap layers can be calculated by

$$\frac{\partial L^{(k')}}{\partial \mathbf{U}} = \begin{bmatrix} \frac{\partial L^{(k+1)}}{\partial \mathbf{X}_k} & \mathbf{0} \end{bmatrix}, \quad (20)$$

$$\frac{\partial L^{(k')}}{\partial \mathbf{\Sigma}} = \mathbf{0}, \quad (21)$$

where $\mathbf{0}$ in Eqn.20 is the matrix of size $D \times (D - q)$ with all elements being zero.

5. Empirical Evaluation

In this section we study the effectiveness of the proposed GrNet structure. In particular, we present results for three typical visual classification tasks being emotion recognition, action recognition and face verification. In the experiments, we compare four state-of-the-art Grassmann learning methods: Discriminative Canonical Correlations (DCC) [39], Grassmann Discriminant Analysis (GDA) [20], Grassmannian Graph-Embedding Discriminant Analysis (GGDA) [19] and Projection Metric Learning (PML) [31] methods. For all of them, we use their source codes from authors with tuning their parameters as in the original papers. For the proposed GrNet, we build its architecture with one or multiple block(s) illustrated in Fig.1 before three output layers, that are ProjMap, FC and softmax layers. The learning rate λ is fixed as 0.01, the batch size is set to 30. Note that multiple projections per FRMap layer (M-FRMap) and matrix element-based ProjPooling (E-ProjPooling) are studied first before the discussion where other configurations are compared. The projection matrices per FRMap layer are initialized as random full rank matrices, and the number of them per layer is set to 16. For all the ProjPooling layers, the number of the instances for pooling are fixed as 4. For training the GrNet, we just use an i7-2600K (3.40GHz) PC without any GPUs.

5.1. Emotion Recognition

For the task of emotion recognition, we employ the popular Acted Facial Expression in Wild (AFEW) [14] dataset. The AFEW dataset contains a dynamic temporal facial expressions data in close to real world setting. It collects 1,345 video sequences of facial expressions acted by 330 actors.

The standard protocol [14] splits the dataset into three data sets, i.e., training, validation and test data sets. In the training and validation data sets, each video is classified into one of seven expressions, while the ground truth of the test set has not been released. As a result, we follow [41, 29] to present the results on the validation set. As done in many works such as [29] for augmenting the training data, we split the training videos to 1,747 small subvideos. For the evaluation, each facial frame is normalized to an image of size 20×20 . Then, following [42, 43], we mode each sequence of facial expression with a linear subspace of order 10. With this setting, the linear subspace representations of the video samples span a Grassmann manifold $\mathcal{G}(10, 400)$.

In the task, the sizes of the GrNet-1Block weights are set to 400×100 , while those of the GrNet-2Blocks are set to 400×300 and 150×100 . Training the GrNet per epoch (100

Method	AFEW	HDM05
STM-ExpLet [41]	31.73%	—
DeepO2P [33]	28.54%	—
RSR-SPDML [21]	—	$48.01\% \pm 3.38$
SPDNet [29]	34.23%	$61.45\% \pm 1.12$
DCC [39]	25.78%	$41.74\% \pm 1.92$
GDA [20]	29.11%	$46.25\% \pm 2.71$
GGDA [19]	29.45%	$46.87\% \pm 2.19$
PML [31]	28.98%	$47.25\% \pm 2.78$
GrNet-0Block	25.34%	$51.12\% \pm 3.55$
GrNet-1Block	32.08%	$57.73\% \pm 2.24$
GrNet-2Blocks	34.23%	$59.23\% \pm 1.78$

Table 1. Emotion/action recognition accuracies for the AFEW and HDM05 database.

epoches in total) costs around 10 minutes(m). As shown in Table.1, we compare the related methods including the state-of-the-art methods (i.e., STM-ExpLet [41], DeepO2P [33] and SPDNet [29]) on this database. The results show the proposed GrNet with 2 blocks significant outperforms the existing Grassmann learning methods, and is comparable with the state-of-the-art methods although the training data is small. As studied in existing Grassmann learning methods, the GrNet without Riemannian computing (i.e., ProjMap) and without geometry-aware learning (i.e., Re-Orth) perform very badly (17.62% and 26.15% resp.). Besides, the improvement of stacking more GrNet blocks verifies it can learn more discriminative Grassmannian data.

5.2. Action Recognition

For the problem of action recognition, we employ the HDM05 database [47] which is one of the largest-scale skeleton-based human action datasets. The dataset consists of 2,337 sequences of 130 action classes, and provides 3D locations of 31 joints of the subjects.

Following the protocol designed in [29], we conduct 10 random evaluations, each of which randomly selected half of sequences for training and the rest for testing. For data augmentation, the training sequences are divided into around 18,000 small subsequences in each random evaluation. As done in [21, 29], we represent each sequence by a covariance descriptor of size 93×93 , which is computed by the second order statistics of the 3D coordinates of the 31 joints in each frame. Then, we apply SVD on each resulting covariance matrix to get linear subspace of order 10. As a result, this would yield data on a Grassmannian $\mathcal{G}(10, 93)$.

For our GrNet-1Block, the sizes of the connection weights are set to 93×60 , while those of GrNet-2Block

Method	PaSC-Con	PaSC-Han
VGGDeepFace [53]	78.82%	68.24%
DeepO2P [33]	68.76%	60.14%
SPDNet [29]	80.12%	72.83%
DCC [39]	75.83 %	67.04%
GDA [20]	71.38%	67.49%
GGDA [19]	66.71%	68.41%
PML [31]	73.45%	68.32%
GrNet-0Block	68.52%	63.92%
GrNet-1Block	80.15%	72.51%
GrNet-2Blocks	80.52%	72.76%

Table 2. Face verification accuracies for the two experiments (i.e., control and handheld) of the PaSC database.

are fixed as $93 \times 80, 40 \times 30$ respectively. The training time at each of 100 epoches is around 9m on average. Table.1 gives the performances of the comparative algorithms and of the state-of-the-art methods (RSR-SPDML [21] and SPD-Net [29]) on this dataset. We can see that our GrNet outperforms the state-of-the-art Grassmann learning methods by a large margin (more than 11%). This verifies that the proposed deep Grassmann learning scheme yields great improvements when the training data is large enough, and again validates the benefits of using more GrNet blocks.

5.3. Face Verification

For face verification, we use one standard video face dataset being Point-and-Shoot Challenge (PaSC) [4]. It contains 2,802 videos of 265 subjects, two halves of which are taken by control and handheld cameras respectively.

For the dataset, [4] designs control and handheld face verification experiments, where the identity in the query video is verified by comparing with the target video. As done in [5, 29], we use its 280 training videos and the external training data (COX) [28] with 900 videos for training. As the last two experiments, the whole training data are also split to 12,529 small subvideos. To extract the state-of-the-art deep face features, we perform the approach of [53] on the normalized face images of size 224×224 . For speeding up the training, we employ PCA to reduce the deep features to 400-dimensional features. Following [32, 29], a SPD matrix of size 401×401 is computed by fusing the data covariance matrix and mean for each video sequence. As done in [31] on each video, we finally compute a linear subspace of order 10, which is on a Grassmann manifold $\mathcal{G}(10, 401)$.

In the evaluation, we configure the sizes of the GrNet weights in FRMap layers to 401×100 in the 1 block case, while setting those to 401×300 and 150×100 in the 2

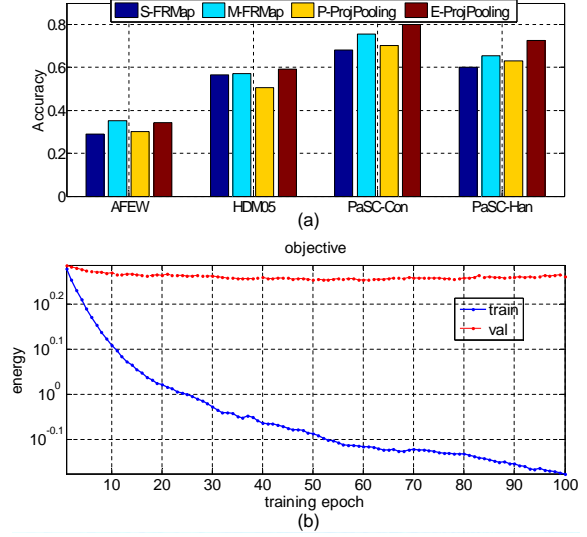


Figure 2. (a) Comparison of single FRMap (S-FRMap), multiple FRMap (M-FRMap), multiple projection-based ProjPooling (S-ProjPooling) and multiple element-based ProjPooling (P-ProjPooling) for the three used databases. (b) Convergence behavior of the proposed GrNet for the AFEW dataset.

block case. The time for training the GrNet at each of 100 epoches is about 13m. Table.1 compares the accuracies of the different methods including the state-of-the-art methods (VGGDeepFace [53], DeepO2P [33] and SPDNet [29]) on the PaSC database. Although the used softmax output layer in the GrNet do not suit the verification task well, we find that it still reaches the highest performances in the case of 2 blocks, which learns more favorable Grassmannian data.

5.4. Discussion

For the manifolds of SPD matrices, [29] developed a deep network (SPDNet), which has shown its superiority over shallow SPD matrix learning scheme as well as recent deep learning methods (e.g., DeepO2P [33]) that work on the Euclidean forms of manifold data. In this paper, the proposed GrNet acts as a deep learning tool on Grassmann manifolds. Obviously, different from SPDNet, it designs the layers specially for Grassmannian data. Besides, the GrNet optimizes the connection weights on PSD manifolds rather than Stiefel manifolds because the former could get better solutions. To validate this, we compare them on the three used datasets, where the Stiefel optimization often achieves worse results (32.13%, 57.25%, 80.15%, 71.28%). Moreover, the GrNet also exploits single and multiple projections per FRMap layer (i.e., S-FRMap, M-FRMap) simultaneously, and devises pooling layer on multiple projections (P-ProjPooling) and multiple elements (E-ProjPooling). Note

that the ProjPooling works together with M-FRMap. These properties also considerably differ from existing Grassmann shallow learning methods e.g., PML method [31]. As presented in Fig.2 (a), for the three used datasets, M-FRMap typically beats S-FRMap, and E-ProjPooling outperforms P-ProjPooling. Finally, we also study that the GrNet is able to converge after hundreds of epochs as shown in Fig.2 (b).

6. Conclusion

Grassmannian data are becoming popular in computer vision community. Motivated by this, in this paper we introduced Grassmann networks, the first deep architecture that performs deep learning over Grassmann manifolds. Essentially, it is a natural exploration of classical deep convolutional neural networks to perform global convolution, normalization, pooling and Riemannian computing on Grassmannian data. In three typical visual classification evaluations, the proposed network significantly outperformed existing Grassmann learning methods, and performed comparably with state-of-the-art methods. Directions for future work include extending to more general Riemannian manifolds (e.g., Stiefel manifolds) and applying to other computer vision problems (e.g., image restoration).

Appendix

Gradient Computation of QR Decomposition

For the QR operation, we differentiate its implicit system

$$0 = \mathbf{X}_{k-1} - \mathbf{Q}\mathbf{R}, \quad (22)$$

$$0 = \mathbf{Q}^T \mathbf{Q} - \mathbf{I}, \quad (23)$$

$$0 = \mathbf{R}_{tril}, \quad (24)$$

where \mathbf{R}_{tril} returns the elements below the main diagonal of \mathbf{R} , and obtain

$$0 = d\mathbf{X}_{k-1} - d\mathbf{Q}\mathbf{R} - \mathbf{Q}d\mathbf{R}, \quad (25)$$

$$0 = d\mathbf{Q}^T \mathbf{Q} + \mathbf{Q}^T d\mathbf{Q}. \quad (26)$$

which means $\mathbf{Q}^T d\mathbf{Q}$ is antisymmetric. Multiplying Eqn.25 from the left with \mathbf{Q}^T and the right with \mathbf{R}^{-1} derives

$$d\mathbf{R} = \mathbf{Q}^T d\mathbf{X}_{k-1} - \mathbf{Q}^T d\mathbf{Q}\mathbf{R}, \quad (27)$$

$$d\mathbf{Q} = d\mathbf{X}\mathbf{R}^{-1} - \mathbf{Q}d\mathbf{R}\mathbf{R}^{-1}. \quad (28)$$

The multiplication of Eqn.27 from the right with the inverse of \mathbf{R} yields the equation

$$0 = \mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1} - \mathbf{Q}^T d\mathbf{Q}\mathbf{R}\mathbf{R}^{-1} - d\mathbf{R}\mathbf{R}^{-1}. \quad (29)$$

As $(d\mathbf{R}\mathbf{R}^{-1})_{tril} = 0$, from Eqn.29 we derive

$$(\mathbf{Q}^T d\mathbf{Q})_{tril} = (\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{tril}. \quad (30)$$

Since $\mathbf{Q}^T d\mathbf{Q}$ is antisymmetric we have

$$\begin{aligned} \mathbf{Q}^T d\mathbf{Q} &= (\mathbf{Q}^T d\mathbf{Q})_{tril} - ((\mathbf{Q}^T d\mathbf{Q})_{tril})^T \\ &= (\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{tril} - ((\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{tril})^T \\ &= (\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{asym}. \end{aligned} \quad (31)$$

By substituting Eqn.31 into Eqn.27, the gradient of the QR decomposition with respect to \mathbf{R} is derived as Eqn.15. By plugging Eqn.15 into Eqn.28, we can derive the gradient of the QR decomposition with respect to \mathbf{Q} as Eqn.14.

Partial Derivatives of the Loss Function for the ReOrth Layers

At first, we employ some properties of matrix inner product $\mathbf{A} : \mathbf{B} = \text{Tr}(\mathbf{A}^T \mathbf{B})$, that were studied in [34], to derive the following equivalent equation

$$\mathbf{A} : \mathbf{B}_{asym} = \mathbf{A}_{bsym} : \mathbf{B}, \quad (32)$$

where $\mathbf{B}_{asym} = \mathbf{B}_{tril} - (\mathbf{B}_{tril})^T$, $\mathbf{A}_{bsym} = \mathbf{A}_{tril} - (\mathbf{A}^T)_{tril}$, \mathbf{A}_{tril} extracts the elements below the main diagonal of \mathbf{A} .

For the new chain rule for QR decomposition in the ReOrth layers, we then can plug Eqn.14 and Eqn.15 into Eqn.13 to achieve the full derivative

$$\begin{aligned} \frac{\partial L^{(k)}}{\partial \mathbf{X}_{k-1}} : d\mathbf{X}_{k-1} &= \frac{\partial L^{(k')}}{\partial \mathbf{Q}} : d\mathbf{Q} + \frac{\partial L^{(k')}}{\partial \mathbf{R}} : d\mathbf{R} \\ &= \frac{\partial L^{(k')}}{\partial \mathbf{Q}} : S d\mathbf{X}_{k-1} \mathbf{R}^{-1} + \frac{\partial L^{(k')}}{\partial \mathbf{Q}} : \mathbf{Q}(\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{asym} \\ &\quad + \frac{\partial L^{(k')}}{\partial \mathbf{R}} : \mathbf{Q}^T d\mathbf{X}_{k-1} - \frac{\partial L^{(k')}}{\partial \mathbf{R}} : (\mathbf{Q}^T d\mathbf{X}_{k-1} \mathbf{R}^{-1})_{asym} \mathbf{R} \\ &= \left(S^T \frac{\partial L^{(k')}}{\partial \mathbf{Q}} + \mathbf{Q} \left(\mathbf{Q}^T \frac{\partial L^{(k')}}{\partial \mathbf{Q}} \right)_{bsym} \right) (\mathbf{R}^{-1})^T : d\mathbf{X}_{k-1} \\ &\quad + \mathbf{Q} \left(\frac{\partial L^{(k')}}{\partial \mathbf{R}} - \left(\frac{\partial L^{(k')}}{\partial \mathbf{R}} \mathbf{R}^T \right)_{bsym} (\mathbf{R}^{-1})^T \right) : d\mathbf{X}_{k-1} \end{aligned} \quad (33)$$

With Eqn.33, the partial derivatives in Eqn.16 of the loss functions for the ReOrth layer can be derived.

References

- [1] P. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica*, 80(2):199–220, 2004.
- [2] P. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.
- [3] M. Bai, W. Luo, K. Kundu, and R. Urtasun. Deep semantic matching for optical flow. *CVPR*, 2016.
- [4] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Given, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, et al. The challenge of face recognition from digital point-and-shoot cameras. In *BTAS*, 2013.
- [5] J. R. Beveridge, H. Zhang, et al. Report on the FG 2015 video person recognition evaluation. In *FG*, 2015.
- [6] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Auto. Control*, 58(9):2217–2229, 2013.
- [7] S. Bonnabel and R. Sepulchre. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1055–1070, 2009.
- [8] D. Boscaini, J. Masci, S. Melzi, M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, pages 13–23. Wiley Online Library, 2015.
- [9] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.
- [10] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and deep locally connected networks on graphs. In *ICLR*, 2013.
- [11] H. E. Cetingul and R. Vidal. Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds. In *CVPR*, 2009.
- [12] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *CVPR*, 2009.
- [13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [14] A. Dhall, R. Goecke, J. Joshi, K. Sikka, and T. Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *ICMI*, 2014.
- [15] Y. Dodge and V. Rousson. Multivariate l_1 mean. *Metrika*, 49(2):127–134, 1999.
- [16] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.
- [17] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.
- [18] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, 2011.
- [19] J. Hamm and D. D. Lee. Extended grassmann kernels for subspace-based learning. In *NIPS*, 2008.
- [20] J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *ICML*, 2008.
- [21] M. T. Harandi, M. Salzmann, and R. Hartley. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *ECCV*. 2014.
- [22] M. T. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, and H. Li. Expanding the family of Grassmannian kernels: An embedding perspective. In *ECCV*. 2014.
- [23] M. T. Harandi, C. Sanderson, C. Shen, and B. Lovell. Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution. In *ICCV*, 2013.
- [24] M. T. Harandi, C. Sanderson, S. Shirazi, and B. C. Lovell. Graph embedding discriminant analysis on Grassmannian manifolds for improved image set matching. In *CVPR*, 2011.
- [25] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.
- [26] U. Helmke, K. Hüper, and J. Trumpf. Newton’s method on Grassmann manifolds. *arXiv preprint arXiv:0709.2205*, 2007.
- [27] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.
- [28] Z. Huang, S. Shan, R. Wang, H. Zhang, S. Lao, A. Kuerban, and X. Chen. A benchmark and comparative study of video-based face recognition on cox face database. *IEEE T-IP*, 24(12):5967–5981, 2015.
- [29] Z. Huang and L. Van Gool. A Riemannian network for SPD matrix learning. *arXiv preprint arXiv:1608.04233*, 2016.
- [30] Z. Huang, R. Wang, X. Li, W. Liu, S. Shan, L. Van Gool, and X. Chen. Geometry-aware similarity learning on SPD manifolds for visual recognition. *arXiv preprint arXiv:1608.04914*, 2016.

- [31] Z. Huang, R. Wang, S. Shan, and X. Chen. Projection metric learning on Grassmann manifold with application to video based face recognition. In *CVPR*, 2015.
- [32] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen. Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *ICML*, 2015.
- [33] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015.
- [34] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *arXiv:1509.07838*, 2016.
- [35] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-RNN: Deep learning on spatio-temporal graphs. In *CVPR*, 2016.
- [36] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.
- [37] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. T. Harandi. Kernel methods on Riemannian manifolds with Gaussian RBF kernels. *IEEE-TPAMI*, 2015.
- [38] M. Journée, F. Bach, P. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIOPT*, 2010.
- [39] T.-K. Kim, J. Kittler, and R. Cipolla. Discriminative learning and recognition of image set classes using canonical correlations. *IEEE T-PAMI*, 29(6):1005–1018, 2007.
- [40] H. Le. On geodesics in Euclidean shape spaces. *J. Lond. Math. Soc.*, pages 360–372, 1991.
- [41] M. Liu, S. Shan, R. Wang, and X. Chen. Learning expressionlets on spatio-temporal manifold for dynamic facial expression recognition. In *CVPR*, 2014.
- [42] M. Liu, R. Wang, Z. Huang, S. Shan, and X. Chen. Partial least squares regression on Grassmannian manifold for emotion recognition. In *ICMI*, 2013.
- [43] M. Liu, R. Wang, S. Li, S. Shan, Z. Huang, and X. Chen. Combining multiple kernel methods on Riemannian manifold for emotion recognition in the wild. In *ICMI*, 2014.
- [44] T. Marrinan, J. Ross Beveridge, B. Draper, M. Kirby, and C. Peterson. Finding the subspace mean or median to fit your need. In *CVPR*, 2014.
- [45] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *ICCV Workshops*, 2015.
- [46] G. Meyer, S. Bonnabel, and R. Sepulchre. Regression on fixed-rank positive semidefinite matrices: a Riemannian approach. *JMLR*, 12:593–625, 2011.
- [47] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation: Mocap database HDM05. *Tech. Rep. CG-2007-2*, 2007.
- [48] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [49] A. Srivastava and E. Klassen. Monte carlo extrinsic estimators of manifold-valued parameters. *IEEE Transactions on Signal Processing*, 50(2):299–308, 2002.
- [50] A. Srivastava and E. Klassen. Bayesian and geometric subspace tracking. *Advances in Applied Probability*, pages 43–56, 2004.
- [51] R. Subbarao and P. Meer. Nonlinear mean shift over Riemannian manifolds. *IJCV*, 84(1):1–20, 2009.
- [52] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE-TPAMI*, 33(11):2273–2286, 2011.
- [53] O. P. A. Vedaldi and A. Zisserman. Deep face recognition. In *BMVC*, 2015.
- [54] R. Vemulapalli, J. Pillai, and R. Chellappa. Kernel learning for extrinsic classification of manifold features. In *CVPR*, 2013.
- [55] Y.-C. Wong. Differential geometry of Grassmann manifolds. *Proceedings of the National Academy of Sciences of the United States of America*, 57(3):589, 1967.
- [56] X. Zeng, W. Bian, W. Liu, J. Shen, and D. Tao. Dictionary pair learning on Grassmann manifolds for image denoising. *IEEE-TIP*, 24(11):4556–4569, 2015.